RADIOSITY

Introduzione

La Radiosity è una tecnica avanzata di Illuminazione Globale, view-independent per il rendering di componenti di luce diffusa.
La premessa dell'algoritmo è semplice: qualsiasi luce che colpisce una superficie viene riflessa nella scena, indipendentemente dalla sorgente da cui essa proviene (luce o una componente di rimbalzo), e questa componente riflessa interagisce con le altre superfici successivamente.
L'algoritmo di Radiosity fa capo a una branca della fisica, la Radiometria, dove la radiosità è il flusso radiante che esce (emesso, riflesso e trasmesso) da una superficie per unità di area, e l'algoritmo applica questo concetto a una scena 3D, dove ogni superficie rilascia una determinata quantità di "Energia". Il punto sta nel determinare la quantità approssimata di energia che una superficie emana una volta colpita da una sorgente di luce.

Proprietà e funzionamento

Dal punto di vista dell'algoritmo, non esiste distinzione fra un oggetto normale e una luce propria, in quanto ogni oggetto dal suo punto di vista ha la potenzialità di essere una fonte di luce.
Il funzionamento è il seguente: ogni superficie che appartiene alla scena viene suddivisa in Patches, per ridurre la scala del problema da grande problema a molteplici piccoli problemi.
Ora, prendiamo 2 patch distinte A e B, prese a piacere all'interno di una scena. La Radiosity si occupa di **determinare quanta energia o radiazione viene trasmessa dalla patch A alla patch B**.
Per fare questo, dobbiamo innanzitutto comprendere quanto bene A e B possono interagire fra di loro, o quanto bene riescono a "vedersi" l'una con l'altra. Questo concetto è espresso nell'algoritmo tramite il coefficiente Form Factor (F_{ij}): il F_{ij} diminuisce quanto più 2 Patch sono distanti o hanno angoli incompatibili fra di loro, con eventuali altre patch disposte in mezzo che le occludono, e portano il valore essenzialmente a 0.

Il calcolo della Radiosity da una patch A a una patch B è ovviamente il risultato di una singola iterazione.
Cosi come algoritmi come il Ray-Tracing, sono necessarie più iterazioni per far si che si prosegua dalla patch B ad altre patch compatibili, e non sorprende che costo computazionale cresca le più interazioni vengono fatte.
In molte scene, un limite di 3-4 iterazioni è sufficiente, siccome la radiazione che lascia una superficie decresce ad ogni rimbalzo, fino a diventare impercettibile all'occhio umano, e ulteriori processi risulterebbero superflui.

Questo processo porta a una serie di proprietà interessanti:

- **Color bleeding:** è un fenomeno che si verifica nel momento in cui la luce rimbalza sulla superficie di un oggetto, e il secondo oggetto, su cui la luce arriva, assume le sfumature di colore del primo. È un effetto di pura illuminazione globale.
- **Soft shadows:** le ombre vengono proiettare con un livello di antialiasing che causa la sfumatura nei bordi.

Analisi dell'algoritmo

Se le superfici vengono approssimate da un numero finito di patch planari, ciascuna delle quali si assume abbia Radiosity costante B_i e riflessività ρ_i per ogni patch i, l'equazione della radiosità discreta è:

$$B_i = E_i + \rho_i \sum_{j=1}^{n} F_{ij} B_j$$

dove:
- F_{ij} è il Form Factor geometrico per la radiazione che esce da j e colpisce la patch i.
- E_i è la capacità di un oggetto (in questo caso la fonte di luce) di emettere energia, detta anche Emissività.
- $SumF_{ij}B_j$ è la sommatoria della luce che incide sulla Patch i da tutte le altre n patch.

Data una scena da renderizzare, si ha che il 100% dell'energia emessa da ciascun elemento deve arrivare agli altri elementi, che in termini matematici indica che:

$$\sum_{j=1}^{n} F_{ij} = 1$$

Questo algoritmo tuttavia ha un peso computazionale considerevole: date n patch nella scena, l'algoritmo presenta una complessità di ($O(n^2)$), quindi è comunque un processo di rendering più oneroso della semplice rasterizzazione.

Per ovviare in parte a ciò, si è sviluppato un metodo per la risoluzione dell'algoritmo detto Progressive Refinement Radiosity: sfrutta la medesima equazione, ma non la risolve tutta immediatamente ma attraverso varie iterazioni, dove viene scelta una patch j (lo "shooter") e usa la sua radiosità attuale per aggiornare le altre.
Accumula gradualmente i risultati, perfezionando la soluzione.
Questo procedimento numerico ha una complessità di O(n*s),dove s sono il numero di iterazioni.
È bene specificare però che s è sempre più piccolo di n: s sceglie come shooter per l'interazione corrente sempre quello che fornisce più energia, per poi arrivare a

valori sempre meno significativi con ogni successiva aggiunta, fino ad arrivare a un certo punto nel quale tutti i successivi shooter hanno un valore troppo piccolo per essere intelligibile. Quindi, vengono scartati.

N.B L'equazione considera solo una luce **monocromatica**. La Radiosity **modella esplicitamente la luce riflessa diffusa**, e la luce diffusa, in quanto la luce incidente sulla superficie, prende il colore della patch stessa.

Come vi calcola il Form Factor?
L'emissività è data come costante in base al materiale che si vuole rappresentare
Dati 2 elementi i e j , il Form Factor può essere calcolato nel seguente modo: si prendono 2 elementi infinitesimi dAj e dAi sulle rispettive superfici, si proiettano sui rispettivi angoli θi e θj rispetto al segmento che li congiunge. In aggiunta, è necessario che le 2 superfici siano VISIBILI fra di loro, quindi si introduce la funzione di visibilità H_{ij} che vale 1 o 0 se le 2 superfici sono visibili o non rispettivamente.
Facendo la cosiddetta media integrale, il Form factor risulta:

$$F_{ij} = \frac{1}{A_i} \int_{A_i} \int_{A_j} \frac{\cos\theta_i \cos\theta_j}{\pi r^2} H_{ij}\, dA_j\, dA_i$$

Possiamo risolvere queste equazioni utilizzando un sistema lineare.

Tuttavia, l'utilizzo di un sistema lineare come questo, sfruttando per esempio il metodo di eliminazione Gaussiana, è troppo costoso per applicazioni reali, poiché richiede un tempo di esecuzione nell'ordine di $O(n^3)$.

Dobbiamo ricorrere a delle **approssimazioni** per calcolare il F_{ij}. 2 metodi che sono stati ideati sono entrambi legati alla **proiezione** della superficie Aj e delle parti infinitesime della superficie: la proiezione su una emisfera e su un emicubo.
Per motivi di efficienza, si pensa che l'approssimazione per emicubo sia quella più comoda quindi viene spesso utilizzata.
Si basa su un'osservazione fondamentale: se poniamo un emicubo (mezzo cubo) sopra una patch i, possiamo usare un rendering z-buffer per stimare quanto di ogni altra patch è "visibile" da i. Analizziamo la scena completa fin dall'inizio.

Un emicubo di lato 1 unità viene posizionato al centro di una patch di cui si vogliono calcolare i Form Factor. Ciascuna delle cinque facce esterne del emicubo è divisa regolarmente in un insieme di patch quadrate chiamate **pixel**. Maggiore è la dimensione dei pixel del emicubo, peggiore sarà la stima dei Form Factor, ma più veloce sarà l'algoritmo, quindi conviene fare un bilanciamento.
Proiettando la patch j sulla superficie, basandosi sulla direzione del vettore della luce riflessa, si ottiene un approssimazione del F_{ij}. Questa proiezione includerà un certo quantitativo di pixel. Questo procedimento, come accennato prima, viene effettuato mediante la tecnologia di **z-buffer**.

L'algoritmo risulta: $\Delta F_p = \frac{\cos\theta_i \cos\theta_p}{\pi r^2}\Delta A$

Dove:

- θ_p è l'angolo tra la normale alla cella p e il vettore r che congiunge il centro di dAi (cioè il centro del cubo) col centro di p.
- θ_i è l'angolo di incidenza.
- ΔA è l'area della cella.

Tuttavia è bene notare che, per un Emicubo, i vari pixel sono orientati in 2 soli modi: quelli appartenenti alla superficie up, e quelli appartenenti alle 4 laterali.
Facendo i calcoli geometrici, si ottiene che $r = \sqrt{1 + x_p^2 + y_p^2}$, con x_p e y_p le coordinate del centro, e che il prodotto dei 2 coseni equivale a 1/r per la patch soprastante, mentre vale z_p/r per quelle laterali.

Quindi, sostituendo nell'equazione, si identificano questi 2 casi:

Riflessione su un pixel soprastante: $\Delta F_p = \frac{1}{\pi(1 + x_p^2 + y_p^2)^2}\,\Delta A$

Riflessione su un pixel laterale: $\Delta F_p = \frac{z_p}{\pi(1 + x_p^2 + y_p^2)^2}\,\Delta A$

Per ogni pixel determiniamo in anticipo il suo form factor in questo modo.
Va infine sommato il contributo di tutti i pixel N su cui viene proiettata la patch j per ottenere il F_{ij} complessivo:

$$F_{ij} = \sum_{n=1}^{N} \Delta F_n$$

Cosi, il tempo totale di esecuzione è approssimative ad $O(n^2)$.

Confronto ed Interazioni con il Ray-tracing

Questi 2 algoritmi a prima vista possono sembrare simili, potendo essere entrambi usati per la rappresentazione dell'illuminazione globale all'interno di una scena.

La principale differenza sta nella struttura stessa: il Ray tracing è una tecnica basata sulla rifrazione della luce speculare, e per questo è view-dependent, e il riflesso viene calcolato partendo dall'occhio dello spettatore calcolando il vettore di rifrazione e arrivando alla sorgente di luce; la Radiosity invece agisce sulle componenti diffusive, perciò è view-indipendent.

Inoltre, il Ray tracing applicato all'illuminazione globale presenta alcune problematiche, in quanto ignora la componente diffusa dell'illuminazione incidente.

Questo causa elementi come, per esempio, le ombre coi margini solidi . Se volessi simulare la componente diffusiva col Ray tracing, sarebbe necessario inviare raggi da ogni punto della superficie all'intero emisfero visibile.
Anche se si potesse calcolare un problema così imponente, esiste un problema concettuale con i loop: date 2 superfici parallele, se il punto A riceve luce dal punto B, anche il punto B riceve luce dal punto A. Con un algoritmo di ray tracing standard, questo richiede una grande quantità di risorse, poiché è necessario emettere un numero enorme di raggi. Per questo motivo, si preferisce la Radiosity. La radiosity invece semplicemente non è progettata per gestire il punto di vista dell'osservatore, quindi non ha senso usare questo metodo per le componenti speculari.

Detto ciò, non è difficile immaginare che queste 2 tecniche posso essere combinate in una singola scena.
Purtroppo però non è sufficiente applicare entrambi i metodi e poi sommare i risultati, perché le componenti della luminosità dovute alla riflessione speculare contribuiscono anche alla illuminazione diffusa e viceversa. In un articolo del 1987 scritto da J.R.Wallace, M.F.Cohen e D.P.Greenberg, viene sviluppato un approccio a due passi:

1) Radiosity ampliata, la quale tiene conto anche della riflessione speculare, così da ampliare in maniera virtuale la scena.
2) Ray tracing ampliato: Invece di un raggio riflessivo che arriva agli occhi dello spettatore, si costruisce una *piramide di riflessione* con apertura piccola. Sul piano perpendicolare all'asse della piramide (cioè alla direzione di riflessione speculare) si considera un piccolo rettangolo che viene usato per eseguire uno z-buffer di piccole dimensioni: così si determina quali altri elementi siano visibili attraverso ciascun pixel, e di ciascuno di essi si considera l'illuminazione diffusa calcolata nella prima fase tramite la Radiosity ampliata (la radiosità è stata riportata ai vertici della scena).

Non è difficile immaginare che questo sia un algoritmo oneroso. Per ovvie ragioni, non è adatto per renderizzare scene dinamiche.

Nel corso degli anni, è stato introdotto anche il Path Tracing, una tecnica avanzata di rendering, multicampionata e randomizzata del ray tracing che è in grado di simulare la Global Illumination e le componenti diffuse. Tuttavia, rimane un algoritmo costoso, in quanto anche alcuni effetti quali il color bleeding necessitano di parecchi calcoli rispetto alla singola Radiosity.